**Exercise 1: Create a Database and Tables**

**Exercise:**

1. Create a database named company\_db.
2. Create a table named employees with the following columns:
   * emp\_id (INT, Primary Key, Auto Increment)
   * first\_name (VARCHAR(50), Not Null)
   * last\_name (VARCHAR(50), Not Null)
   * department\_id (INT)
   * salary (DECIMAL(10, 2), Not Null)
3. Create a table named departments with the following columns:
   * department\_id (INT, Primary Key, Auto Increment)
   * department\_name (VARCHAR(50), Not Null, Unique)

**Solution:**

-- 1. Create database

CREATE DATABASE company\_db;

USE company\_db;

-- 2. Create employees table

CREATE TABLE employees (

emp\_id INT AUTO\_INCREMENT PRIMARY KEY,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

department\_id INT,

salary DECIMAL(10, 2) NOT NULL

);

-- 3. Create departments table

CREATE TABLE departments (

department\_id INT AUTO\_INCREMENT PRIMARY KEY,

department\_name VARCHAR(50) NOT NULL UNIQUE

);

**Exercise 2: Alter Table by Adding Columns**

**Exercise:**

1. Add a email column to the employees table with VARCHAR(100) and make it unique.
2. Add a hire\_date column to the employees table as DATE.
3. Modify the salary column to be DECIMAL(12, 2).

**Solution:**

-- 1. Add email column with a unique constraint

ALTER TABLE employees

ADD COLUMN email VARCHAR(100) UNIQUE;

-- 2. Add hire\_date column

ALTER TABLE employees

ADD COLUMN hire\_date DATE;

-- 3. Modify salary column to increase its precision

ALTER TABLE employees

MODIFY COLUMN salary DECIMAL(12, 2);

**Exercise 3: Add Foreign Key Constraints**

**Exercise:**

1. Add a foreign key constraint from employees.department\_id to departments.department\_id.
2. Ensure that when a department is deleted, employees from that department are set to NULL for their department\_id.

**Solution:**

-- Add foreign key constraint

ALTER TABLE employees

ADD CONSTRAINT fk\_department

FOREIGN KEY (department\_id) REFERENCES departments(department\_id)

ON DELETE SET NULL;

**Exercise 4: Rename Tables and Columns**

**Exercise:**

1. Rename the employees table to company\_employees.
2. Rename the first\_name and last\_name columns in the company\_employees table to fname and lname.

**Solution:**

-- 1. Rename table

ALTER TABLE employees RENAME TO company\_employees;

-- 2. Rename columns first\_name and last\_name

ALTER TABLE company\_employees

CHANGE COLUMN first\_name fname VARCHAR(50),

CHANGE COLUMN last\_name lname VARCHAR(50);

**Exercise 5: Drop a Column and a Foreign Key**

**Exercise:**

1. Drop the email column from the company\_employees table.
2. Drop the foreign key constraint that links department\_id between company\_employees and departments.

**Solution:**

-- 1. Drop email column

ALTER TABLE company\_employees

DROP COLUMN email;

-- 2. Drop foreign key constraint

ALTER TABLE company\_employees

DROP FOREIGN KEY fk\_department;

**Exercise 6: Create an Index and Drop It**

**Exercise:**

1. Create an index on the lname column of the company\_employees table.
2. Drop the created index on the lname column.

**Solution:**

-- 1. Create index on lname column

CREATE INDEX idx\_lname ON company\_employees(lname);

-- 2. Drop the index

DROP INDEX idx\_lname ON company\_employees;

**Exercise 7: Truncate and Drop Tables**

**Exercise:**

1. Truncate the departments table.
2. Drop the departments table.

**Solution:**

-- 1. Truncate departments table

TRUNCATE TABLE departments;

-- 2. Drop departments table

DROP TABLE departments;

**Exercise 8: Alter a Table with Multiple Changes**

**Exercise:**

1. In the company\_employees table, add a new column manager\_id as INT.
2. Add a foreign key on manager\_id that references company\_employees(emp\_id).
3. Modify the hire\_date column to DATETIME.

**Solution:**

-- 1. Add manager\_id column

ALTER TABLE company\_employees

ADD COLUMN manager\_id INT;

-- 2. Add foreign key constraint on manager\_id

ALTER TABLE company\_employees

ADD CONSTRAINT fk\_manager

FOREIGN KEY (manager\_id) REFERENCES company\_employees(emp\_id);

-- 3. Modify hire\_date to DATETIME

ALTER TABLE company\_employees

MODIFY hire\_date DATETIME;

**Exercise 9: Drop Multiple Tables at Once**

**Exercise:**

1. Drop the company\_employees and departments tables if they exist.

**Solution:**

DROP TABLE IF EXISTS company\_employees, departments;

**Exercise 10: Add a CHECK Constraint and Drop It**

**Exercise:**

1. Add a CHECK constraint to the company\_employees table that ensures salary is greater than 0.
2. Drop the CHECK constraint from the salary column.

**Solution:**

-- 1. Add CHECK constraint

ALTER TABLE company\_employees

ADD CONSTRAINT check\_salary CHECK (salary > 0);

-- 2. Drop CHECK constraint

ALTER TABLE company\_employees

DROP CONSTRAINT check\_salary;

**Exercise 11: Rename Multiple Tables at Once**

**Exercise:**

1. Rename the company\_employees table to employees, and rename the departments table to company\_departments in a single query.

**Solution:**

RENAME TABLE company\_employees TO employees, departments TO company\_departments;

**Exercise 12: Create a Table with Composite Keys**

**Exercise:**

1. Create a table named project\_assignments with the following columns:
   * emp\_id (INT, Foreign Key to employees(emp\_id))
   * project\_id (INT, Not Null)
   * assigned\_date (DATE, Not Null)
   * Composite Primary Key on emp\_id and project\_id

**Solution:**

CREATE TABLE project\_assignments (

emp\_id INT,

project\_id INT NOT NULL,

assigned\_date DATE NOT NULL,

PRIMARY KEY (emp\_id, project\_id),

FOREIGN KEY (emp\_id) REFERENCES employees(emp\_id)

);

**Exercise 13: Add Multiple Constraints and Indexes**

**Exercise:**

1. In the employees table, add a column phone\_number (VARCHAR(15)) and ensure it is unique.
2. Add a CHECK constraint that ensures the salary is at least 30000.
3. Add an index on the hire\_date column.

**Solution:**

-- 1. Add phone\_number column with unique constraint

ALTER TABLE employees

ADD COLUMN phone\_number VARCHAR(15) UNIQUE;

-- 2. Add CHECK constraint on salary

ALTER TABLE employees

ADD CONSTRAINT check\_salary\_min CHECK (salary >= 30000);

-- 3. Add index on hire\_date

CREATE INDEX idx\_hire\_date ON employees(hire\_date);

**Exercise 14: Modify and Rename Multiple Columns**

**Exercise:**

1. In the employees table, rename the column phone\_number to contact\_number.
2. Modify the contact\_number column to be VARCHAR(20).
3. Rename the hire\_date column to start\_date and change its data type to DATETIME.

**Solution:**

-- 1. Rename and modify contact\_number

ALTER TABLE employees

CHANGE COLUMN phone\_number contact\_number VARCHAR(20);

-- 2. Rename hire\_date to start\_date and change data type

ALTER TABLE employees

CHANGE COLUMN hire\_date start\_date DATETIME;

**Exercise 15: Create Tables with Foreign Key Constraints**

**Exercise:**

1. Create a projects table with the following columns:
   * project\_id (INT, Primary Key, Auto Increment)
   * project\_name (VARCHAR(100), Not Null)
   * start\_date (DATE, Not Null)
   * end\_date (DATE)
2. Create a project\_assignments table with the following columns:
   * assignment\_id (INT, Primary Key, Auto Increment)
   * emp\_id (INT, Foreign Key references employees(emp\_id))
   * project\_id (INT, Foreign Key references projects(project\_id))
   * role (VARCHAR(50))

**Solution:**

-- 1. Create projects table

CREATE TABLE projects (

project\_id INT AUTO\_INCREMENT PRIMARY KEY,

project\_name VARCHAR(100) NOT NULL,

start\_date DATE NOT NULL,

end\_date DATE

);

-- 2. Create project\_assignments table

CREATE TABLE project\_assignments (

assignment\_id INT AUTO\_INCREMENT PRIMARY KEY,

emp\_id INT,

project\_id INT,

role VARCHAR(50),

FOREIGN KEY (emp\_id) REFERENCES employees(emp\_id),

FOREIGN KEY (project\_id) REFERENCES projects(project\_id)

);

**Exercise 16: Drop Columns and Foreign Keys**

**Exercise:**

1. Drop the contact\_number column from the employees table.
2. Drop the foreign key constraint that references employees(emp\_id) from the project\_assignments table.

**Solution:**

-- 1. Drop contact\_number column

ALTER TABLE employees

DROP COLUMN contact\_number;

-- 2. Drop foreign key constraint on emp\_id in project\_assignments

ALTER TABLE project\_assignments

DROP FOREIGN KEY project\_assignments\_ibfk\_1;

**Exercise 17: Truncate Tables with Foreign Key Constraints**

**Exercise:**

1. Truncate the project\_assignments table.
2. Truncate the projects table.

**Note:** MySQL does not allow truncating tables that are referenced by foreign keys, so you may need to drop the foreign key constraint first.

**Solution:**

-- Drop the foreign key constraint before truncating

ALTER TABLE project\_assignments

DROP FOREIGN KEY project\_assignments\_ibfk\_2;

-- 1. Truncate project\_assignments table

TRUNCATE TABLE project\_assignments;

-- 2. Truncate projects table

TRUNCATE TABLE projects;

**Exercise 18: Rename Multiple Tables and Columns**

**Exercise:**

1. Rename the project\_assignments table to employee\_projects.
2. Rename the role column in the employee\_projects table to job\_role.
3. Rename the projects table to company\_projects.

**Solution:**

-- 1. Rename project\_assignments table to employee\_projects

RENAME TABLE project\_assignments TO employee\_projects;

-- 2. Rename role column to job\_role

ALTER TABLE employee\_projects

CHANGE COLUMN role job\_role VARCHAR(50);

-- 3. Rename projects table to company\_projects

RENAME TABLE projects TO company\_projects;

**Exercise 19: Drop Tables and Constraints**

**Exercise:**

1. Drop the company\_projects table.
2. Drop the employee\_projects table.
3. Drop the check\_salary\_min constraint from the employees table.

**Solution:**

-- 1. Drop company\_projects table

DROP TABLE company\_projects;

-- 2. Drop employee\_projects table

DROP TABLE employee\_projects;

-- 3. Drop CHECK constraint from employees

ALTER TABLE employees

DROP CONSTRAINT check\_salary\_min;

**Exercise 20: Add and Drop Multiple Columns**

**Exercise:**

1. Add the following columns to the employees table:
   * department\_name (VARCHAR(50))
   * job\_title (VARCHAR(50))
2. Drop both department\_name and job\_title columns.

**Solution:**

-- 1. Add multiple columns to employees table

ALTER TABLE employees

ADD COLUMN department\_name VARCHAR(50),

ADD COLUMN job\_title VARCHAR(50);

-- 2. Drop both columns

ALTER TABLE employees

DROP COLUMN department\_name,

DROP COLUMN job\_title;

**Exercise 21: Create a Table with Composite Primary Key and Foreign Key**

**Exercise:**

1. Create a project\_employees table with the following columns:
   * emp\_id (INT, Foreign Key references employees(emp\_id))
   * project\_id (INT, Foreign Key references projects(project\_id))
   * assignment\_date (DATE, Not Null)
   * Primary Key on (emp\_id, project\_id).

**Solution:**

CREATE TABLE project\_employees (

emp\_id INT,

project\_id INT,

assignment\_date DATE NOT NULL,

PRIMARY KEY (emp\_id, project\_id),

FOREIGN KEY (emp\_id) REFERENCES employees(emp\_id),

FOREIGN KEY (project\_id) REFERENCES projects(project\_id)

);

**Exercise 22: Add Check Constraints and Drop Them**

**Exercise:**

1. Add a CHECK constraint on the salary column of the employees table to ensure it’s greater than 50000.
2. Add a CHECK constraint on the assignment\_date in the project\_employees table to ensure the date is not in the future.
3. Drop both CHECK constraints.

**Solution:**

-- 1. Add CHECK constraint on salary

ALTER TABLE employees

ADD CONSTRAINT check\_salary\_above\_50k CHECK (salary > 50000);

-- 2. Add CHECK constraint on assignment\_date

ALTER TABLE project\_employees

ADD CONSTRAINT check\_assignment\_date CHECK (assignment\_date <= CURDATE());

-- 3. Drop CHECK constraints

ALTER TABLE employees

DROP CONSTRAINT check\_salary\_above\_50k;

ALTER TABLE project\_employees

DROP CONSTRAINT check\_assignment\_date;

**Exercise 23: Truncate and Drop Multiple Tables**

**Exercise:**

1. Truncate the project\_employees table.
2. Truncate the employees table.
3. Drop both project\_employees and employees tables.

**Solution:**

-- 1. Truncate project\_employees table

TRUNCATE TABLE project\_employees;

-- 2. Truncate employees table

TRUNCATE TABLE employees;

-- 3. Drop both tables

DROP TABLE project\_employees, employees;

**Exercise 24: Alter Table with Composite Foreign Keys**

**Exercise:**

1. Alter the project\_employees table to add a composite foreign key (emp\_id, project\_id) that references the employees and projects tables, respectively.

**Solution:**

-- Add composite foreign key to project\_employees table

ALTER TABLE project\_employees

ADD CONSTRAINT fk\_composite FOREIGN KEY (emp\_id, project\_id)

REFERENCES employees(emp\_id), projects(project\_id);

**Exercise 25: Alter and Add a Composite Primary Key**

**Exercise:**

1. Create a departments table with the following columns:
   * dept\_id (INT, Primary Key, Auto Increment)
   * dept\_name (VARCHAR(100), Not Null)
2. Create an employees table with the following columns:
   * emp\_id (INT, Auto Increment)
   * emp\_name (VARCHAR(100), Not Null)
   * dept\_id (INT, Foreign Key references departments(dept\_id))
3. Alter the employees table to add a **composite primary key** on both emp\_id and dept\_id.

**Solution:**

-- 1. Create departments table

CREATE TABLE departments (

dept\_id INT AUTO\_INCREMENT PRIMARY KEY,

dept\_name VARCHAR(100) NOT NULL

);

-- 2. Create employees table

CREATE TABLE employees (

emp\_id INT AUTO\_INCREMENT,

emp\_name VARCHAR(100) NOT NULL,

dept\_id INT,

FOREIGN KEY (dept\_id) REFERENCES departments(dept\_id)

);

-- 3. Alter employees table to add composite primary key on emp\_id and dept\_id

ALTER TABLE employees

ADD PRIMARY KEY (emp\_id, dept\_id);

**Exercise 26: Drop and Re-add Foreign Key Constraint**

**Exercise:**

1. Create a projects table with the following columns:
   * project\_id (INT, Primary Key, Auto Increment)
   * project\_name (VARCHAR(100), Not Null)
   * dept\_id (INT, Foreign Key references departments(dept\_id))
2. Alter the projects table to **drop** the foreign key constraint on dept\_id.
3. Re-add the foreign key constraint on dept\_id that references departments(dept\_id) with ON DELETE CASCADE.

**Solution:**

-- 1. Create projects table

CREATE TABLE projects (

project\_id INT AUTO\_INCREMENT PRIMARY KEY,

project\_name VARCHAR(100) NOT NULL,

dept\_id INT,

FOREIGN KEY (dept\_id) REFERENCES departments(dept\_id)

);

-- 2. Drop the foreign key constraint on dept\_id

ALTER TABLE projects

DROP FOREIGN KEY projects\_ibfk\_1;

-- 3. Re-add the foreign key with ON DELETE CASCADE

ALTER TABLE projects

ADD CONSTRAINT fk\_projects\_dept FOREIGN KEY (dept\_id)

REFERENCES departments(dept\_id) ON DELETE CASCADE;

**Exercise 27: Alter Foreign Key and Change Data Type**

**Exercise:**

1. Create a tasks table with the following columns:
   * task\_id (INT, Primary Key, Auto Increment)
   * task\_name (VARCHAR(100), Not Null)
   * emp\_id (INT, Foreign Key references employees(emp\_id))
2. Alter the tasks table to change the data type of emp\_id to BIGINT.
3. Re-add the foreign key constraint to reference the modified column.

**Solution:**

-- 1. Create tasks table

CREATE TABLE tasks (

task\_id INT AUTO\_INCREMENT PRIMARY KEY,

task\_name VARCHAR(100) NOT NULL,

emp\_id INT,

FOREIGN KEY (emp\_id) REFERENCES employees(emp\_id)

);

-- 2. Alter tasks table to change emp\_id data type to BIGINT

ALTER TABLE tasks

MODIFY COLUMN emp\_id BIGINT;

-- 3. Re-add the foreign key after altering the column

ALTER TABLE tasks

ADD CONSTRAINT fk\_tasks\_emp FOREIGN KEY (emp\_id)

REFERENCES employees(emp\_id);

**Exercise 28: Drop Primary Key and Add New Composite Primary Key**

**Exercise:**

1. Create a projects\_tasks table with the following columns:
   * project\_id (INT, Foreign Key references projects(project\_id))
   * task\_id (INT, Foreign Key references tasks(task\_id))
   * assignment\_date (DATE, Not Null)
2. Drop the existing primary key (if any) from the projects\_tasks table.
3. Add a **composite primary key** on both project\_id and task\_id.

**Solution:**

-- 1. Create projects\_tasks table

CREATE TABLE projects\_tasks (

project\_id INT,

task\_id INT,

assignment\_date DATE NOT NULL,

FOREIGN KEY (project\_id) REFERENCES projects(project\_id),

FOREIGN KEY (task\_id) REFERENCES tasks(task\_id)

);

-- 2. Drop the primary key if there is one

-- In this case, assume no primary key was initially created, so skip this step.

-- 3. Add composite primary key on project\_id and task\_id

ALTER TABLE projects\_tasks

ADD PRIMARY KEY (project\_id, task\_id);

**Exercise 29: Alter Foreign Key to Include ON UPDATE CASCADE**

**Exercise:**

1. Create a managers table with the following columns:
   * manager\_id (INT, Primary Key, Auto Increment)
   * manager\_name (VARCHAR(100), Not Null)
   * dept\_id (INT, Foreign Key references departments(dept\_id))
2. Alter the foreign key constraint on dept\_id to add ON UPDATE CASCADE.

**Solution:**

-- 1. Create managers table

CREATE TABLE managers (

manager\_id INT AUTO\_INCREMENT PRIMARY KEY,

manager\_name VARCHAR(100) NOT NULL,

dept\_id INT,

FOREIGN KEY (dept\_id) REFERENCES departments(dept\_id)

);

-- 2. Alter foreign key on dept\_id to add ON UPDATE CASCADE

ALTER TABLE managers

DROP FOREIGN KEY managers\_ibfk\_1;

ALTER TABLE managers

ADD CONSTRAINT fk\_managers\_dept FOREIGN KEY (dept\_id)

REFERENCES departments(dept\_id) ON UPDATE CASCADE;

**Exercise 30: Add and Modify Multiple Foreign Keys**

**Exercise:**

1. Create a tasks\_log table with the following columns:
   * log\_id (INT, Primary Key, Auto Increment)
   * task\_id (INT, Foreign Key references tasks(task\_id))
   * manager\_id (INT, Foreign Key references managers(manager\_id))
   * log\_date (DATE, Not Null)
2. Modify the foreign key on task\_id to include ON DELETE CASCADE.
3. Modify the foreign key on manager\_id to include ON UPDATE CASCADE.

**Solution:**

-- 1. Create tasks\_log table

CREATE TABLE tasks\_log (

log\_id INT AUTO\_INCREMENT PRIMARY KEY,

task\_id INT,

manager\_id INT,

log\_date DATE NOT NULL,

FOREIGN KEY (task\_id) REFERENCES tasks(task\_id),

FOREIGN KEY (manager\_id) REFERENCES managers(manager\_id)

);

-- 2. Modify foreign key on task\_id to add ON DELETE CASCADE

ALTER TABLE tasks\_log

DROP FOREIGN KEY tasks\_log\_ibfk\_1;

ALTER TABLE tasks\_log

ADD CONSTRAINT fk\_tasks\_log\_task FOREIGN KEY (task\_id)

REFERENCES tasks(task\_id) ON DELETE CASCADE;

-- 3. Modify foreign key on manager\_id to add ON UPDATE CASCADE

ALTER TABLE tasks\_log

DROP FOREIGN KEY tasks\_log\_ibfk\_2;

ALTER TABLE tasks\_log

ADD CONSTRAINT fk\_tasks\_log\_manager FOREIGN KEY (manager\_id)

REFERENCES managers(manager\_id) ON UPDATE CASCADE;

**Exercise 31: Add and Drop Foreign Key in a Large Schema**

**Exercise:**

1. Create a workshops table with the following columns:
   * workshop\_id (INT, Primary Key, Auto Increment)
   * workshop\_name (VARCHAR(100), Not Null)
2. Create a workshop\_attendance table with the following columns:
   * attendance\_id (INT, Primary Key, Auto Increment)
   * emp\_id (INT, Foreign Key references employees(emp\_id))
   * workshop\_id (INT, Foreign Key references workshops(workshop\_id))
3. Drop the foreign key on emp\_id from the workshop\_attendance table and re-add it with ON DELETE CASCADE.

**Solution:**

-- 1. Create workshops table

CREATE TABLE workshops (

workshop\_id INT AUTO\_INCREMENT PRIMARY KEY,

workshop\_name VARCHAR(100) NOT NULL

);

-- 2. Create workshop\_attendance table

CREATE TABLE workshop\_attendance (

attendance\_id INT AUTO\_INCREMENT PRIMARY KEY,

emp\_id INT,

workshop\_id INT,

FOREIGN KEY (emp\_id) REFERENCES employees(emp\_id),

FOREIGN KEY (workshop\_id) REFERENCES workshops(workshop\_id)

);

-- 3. Drop and re-add foreign key on emp\_id with ON DELETE CASCADE

ALTER TABLE workshop\_attendance

DROP FOREIGN KEY workshop\_attendance\_ibfk\_1;

ALTER TABLE workshop\_attendance

ADD CONSTRAINT fk\_attendance\_emp FOREIGN KEY (emp\_id)

REFERENCES employees(emp\_id) ON DELETE CASCADE;

**Exercise 32: Add and Drop Composite Foreign Key**

**Exercise:**

1. Create a task\_allocations table with the following columns:
   * allocation\_id (INT, Primary Key, Auto Increment)
   * project\_id (INT, Foreign Key references projects(project\_id))
   * task\_id (INT, Foreign Key references tasks(task\_id))
2. Alter the task\_allocations table to add a **composite foreign key** on both project\_id and task\_id.
3. Drop the composite foreign key.

**Solution:**

-- 1. Create task\_allocations table

CREATE TABLE task\_allocations (

allocation\_id INT AUTO\_INCREMENT PRIMARY KEY,

project\_id INT,

task\_id INT,

FOREIGN KEY (project\_id) REFERENCES projects(project\_id),

FOREIGN KEY (task\_id) REFERENCES tasks(task\_id)

);

-- 2. Add composite foreign key on project\_id and task\_id

ALTER TABLE task\_allocations

ADD CONSTRAINT fk\_task\_allocations\_project\_task FOREIGN KEY (project\_id, task\_id)

REFERENCES projects\_tasks(project\_id, task\_id);

-- 3. Drop the composite foreign key

ALTER TABLE task\_allocations

DROP FOREIGN KEY fk\_task\_allocations\_project\_task;

**Exercise 33: Add a Composite Foreign Key on Two Columns**

**Exercise:**

1. Create a training\_sessions table with the following columns:
   * session\_id (INT, Primary Key, Auto Increment)
   * session\_name (VARCHAR(100), Not Null)
   * trainer\_id (INT, Foreign Key references employees(emp\_id))
2. Create a session\_attendance table with the following columns:
   * attendance\_id (INT, Primary Key, Auto Increment)
   * emp\_id (INT, Foreign Key references employees(emp\_id))
   * session\_id (INT, Foreign Key references training\_sessions(session\_id))
3. Alter the session\_attendance table to add a **composite foreign key** on emp\_id and session\_id.

**Solution:**

-- 1. Create training\_sessions table

CREATE TABLE training\_sessions (

session\_id INT AUTO\_INCREMENT PRIMARY KEY,

session\_name VARCHAR(100) NOT NULL,

trainer\_id INT,

FOREIGN KEY (trainer\_id) REFERENCES employees(emp\_id)

);

-- 2. Create session\_attendance table

CREATE TABLE session\_attendance (

attendance\_id INT AUTO\_INCREMENT PRIMARY KEY,

emp\_id INT,

session\_id INT,

FOREIGN KEY (emp\_id) REFERENCES employees(emp\_id),

FOREIGN KEY (session\_id) REFERENCES training\_sessions(session\_id)

);

-- 3. Add composite foreign key on emp\_id and session\_id

ALTER TABLE session\_attendance

ADD CONSTRAINT fk\_attendance\_emp\_session FOREIGN KEY (emp\_id, session\_id)

REFERENCES employees(emp\_id), training\_sessions(session\_id);

**Exercise 34: Modify Primary Key and Foreign Key on Multiple Columns**

**Exercise:**

1. Create a course\_catalog table with the following columns:
   * course\_id (INT, Primary Key, Auto Increment)
   * course\_name (VARCHAR(100), Not Null)
2. Create a student\_courses table with the following columns:
   * student\_id (INT, Foreign Key references employees(emp\_id))
   * course\_id (INT, Foreign Key references course\_catalog(course\_id))
3. Alter the student\_courses table to drop the current primary key and create a **composite primary key** on student\_id and course\_id.

**Solution:**

-- 1. Create course\_catalog table

CREATE TABLE course\_catalog (

course\_id INT AUTO\_INCREMENT PRIMARY KEY,

course\_name VARCHAR(100) NOT NULL

);

-- 2. Create student\_courses table

CREATE TABLE student\_courses (

student\_id INT,

course\_id INT,

FOREIGN KEY (student\_id) REFERENCES employees(emp\_id),

FOREIGN KEY (course\_id) REFERENCES course\_catalog(course\_id)

);

-- 3. Drop current primary key and add composite primary key on student\_id and course\_id

ALTER TABLE student\_courses

ADD PRIMARY KEY (student\_id, course\_id);

**Exercise 35: Add Multiple Foreign Keys with ON DELETE and ON UPDATE Options**

**Exercise:**

1. Create a clients table with the following columns:
   * client\_id (INT, Primary Key, Auto Increment)
   * client\_name (VARCHAR(100), Not Null)
2. Create a client\_projects table with the following columns:
   * client\_id (INT, Foreign Key references clients(client\_id))
   * project\_id (INT, Foreign Key references projects(project\_id))
3. Alter the client\_projects table to:
   * Add ON DELETE CASCADE to the client\_id foreign key.
   * Add ON UPDATE CASCADE to the project\_id foreign key.

**Solution:**

-- 1. Create clients table

CREATE TABLE clients (

client\_id INT AUTO\_INCREMENT PRIMARY KEY,

client\_name VARCHAR(100) NOT NULL

);

-- 2. Create client\_projects table

CREATE TABLE client\_projects (

client\_id INT,

project\_id INT,

FOREIGN KEY (client\_id) REFERENCES clients(client\_id),

FOREIGN KEY (project\_id) REFERENCES projects(project\_id)

);

-- 3. Modify foreign keys with ON DELETE and ON UPDATE options

ALTER TABLE client\_projects

DROP FOREIGN KEY client\_projects\_ibfk\_1;

ALTER TABLE client\_projects

ADD CONSTRAINT fk\_client\_projects\_client FOREIGN KEY (client\_id)

REFERENCES clients(client\_id) ON DELETE CASCADE;

ALTER TABLE client\_projects

DROP FOREIGN KEY client\_projects\_ibfk\_2;

ALTER TABLE client\_projects

ADD CONSTRAINT fk\_client\_projects\_project FOREIGN KEY (project\_id)

REFERENCES projects(project\_id) ON UPDATE CASCADE;

**Exercise 36: Add and Drop Primary Key with Altered Columns**

**Exercise:**

1. Create a vendors table with the following columns:
   * vendor\_id (INT, Primary Key, Auto Increment)
   * vendor\_name (VARCHAR(100), Not Null)
2. Create a vendor\_products table with the following columns:
   * product\_id (INT, Foreign Key references projects(project\_id))
   * vendor\_id (INT, Foreign Key references vendors(vendor\_id))
3. Alter the vendor\_products table to drop the primary key, modify vendor\_id and product\_id data types to BIGINT, and then re-add the composite primary key on vendor\_id and product\_id.

**Solution:**

-- 1. Create vendors table

CREATE TABLE vendors (

vendor\_id INT AUTO\_INCREMENT PRIMARY KEY,

vendor\_name VARCHAR(100) NOT NULL

);

-- 2. Create vendor\_products table

CREATE TABLE vendor\_products (

product\_id INT,

vendor\_id INT,

FOREIGN KEY (product\_id) REFERENCES projects(project\_id),

FOREIGN KEY (vendor\_id) REFERENCES vendors(vendor\_id)

);

-- 3. Drop primary key, modify columns, and re-add composite primary key

ALTER TABLE vendor\_products

DROP PRIMARY KEY;

ALTER TABLE vendor\_products

MODIFY product\_id BIGINT,

MODIFY vendor\_id BIGINT;

ALTER TABLE vendor\_products

ADD PRIMARY KEY (vendor\_id, product\_id);

**Exercise 37: Rename and Modify Foreign Key Constraints**

**Exercise:**

1. Create a contractors table with the following columns:
   * contractor\_id (INT, Primary Key, Auto Increment)
   * contractor\_name (VARCHAR(100), Not Null)
2. Create a contractor\_assignments table with the following columns:
   * contractor\_id (INT, Foreign Key references contractors(contractor\_id))
   * project\_id (INT, Foreign Key references projects(project\_id))
3. Rename the foreign key on contractor\_id to fk\_contractor\_assignment and add the ON DELETE CASCADE option.

**Solution:**

-- 1. Create contractors table

CREATE TABLE contractors (

contractor\_id INT AUTO\_INCREMENT PRIMARY KEY,

contractor\_name VARCHAR(100) NOT NULL

);

-- 2. Create contractor\_assignments table

CREATE TABLE contractor\_assignments (

contractor\_id INT,

project\_id INT,

FOREIGN KEY (contractor\_id) REFERENCES contractors(contractor\_id),

FOREIGN KEY (project\_id) REFERENCES projects(project\_id)

);

-- 3. Rename foreign key and add ON DELETE CASCADE

ALTER TABLE contractor\_assignments

DROP FOREIGN KEY contractor\_assignments\_ibfk\_1;

ALTER TABLE contractor\_assignments

ADD CONSTRAINT fk\_contractor\_assignment FOREIGN KEY (contractor\_id)

REFERENCES contractors(contractor\_id) ON DELETE CASCADE;

**Exercise 38: Create, Drop, and Re-add Foreign Key with ON UPDATE CASCADE**

**Exercise:**

1. Create a consultants table with the following columns:
   * consultant\_id (INT, Primary Key, Auto Increment)
   * consultant\_name (VARCHAR(100), Not Null)
2. Create a consultant\_tasks table with the following columns:
   * consultant\_id (INT, Foreign Key references consultants(consultant\_id))
   * task\_id (INT, Foreign Key references tasks(task\_id))
3. Alter the consultant\_tasks table to:
   * Drop the current foreign key on consultant\_id.
   * Re-add the foreign key with ON UPDATE CASCADE.

**Solution:**

-- 1. Create consultants table

CREATE TABLE consultants (

consultant\_id INT AUTO\_INCREMENT PRIMARY KEY,

consultant\_name VARCHAR(100) NOT NULL

);

-- 2. Create consultant\_tasks table

CREATE TABLE consultant\_tasks (

consultant\_id INT,

task\_id INT,

FOREIGN KEY (consultant\_id) REFERENCES consultants(consultant\_id),

FOREIGN KEY (task\_id) REFERENCES tasks(task\_id)

);

-- 3. Drop the foreign key and re-add with ON UPDATE CASCADE

ALTER TABLE consultant\_tasks

DROP FOREIGN KEY consultant\_tasks\_ibfk\_1;

ALTER TABLE consultant\_tasks

ADD CONSTRAINT fk\_consultant\_tasks FOREIGN KEY (consultant\_id)

REFERENCES consultants(consultant\_id) ON UPDATE CASCADE;

**Exercise 39: Alter Multiple Foreign Keys Simultaneously**

**Exercise:**

1. Create a meetings table with the following columns:
   * meeting\_id (INT, Primary Key, Auto Increment)
   * meeting\_name (VARCHAR(100), Not Null)
   * client\_id (INT, Foreign Key references clients(client\_id))
2. Create a meeting\_attendees table with the following columns:
   * attendee\_id (INT, Primary Key, Auto Increment)
   * meeting\_id (INT, Foreign Key references meetings(meeting\_id))
   * emp\_id (INT, Foreign Key references employees(emp\_id))
3. Alter the meeting\_attendees table to:
   * Drop both foreign keys on meeting\_id and emp\_id.
   * Re-add both foreign keys with ON DELETE CASCADE for meeting\_id and ON UPDATE CASCADE for emp\_id.

**Solution:**

-- 1. Create meetings table

CREATE TABLE meetings (

meeting\_id INT AUTO\_INCREMENT PRIMARY KEY,

meeting\_name VARCHAR(100) NOT NULL,

client\_id INT,

FOREIGN KEY (client\_id) REFERENCES clients(client\_id)

);

-- 2. Create meeting\_attendees table

CREATE TABLE meeting\_attendees (

attendee\_id INT AUTO\_INCREMENT PRIMARY KEY,

meeting\_id INT,

emp\_id INT,

FOREIGN KEY (meeting\_id) REFERENCES meetings(meeting\_id),

FOREIGN KEY (emp\_id) REFERENCES employees(emp\_id)

);

-- 3. Drop and re-add foreign keys with cascade options

ALTER TABLE meeting\_attendees

DROP FOREIGN KEY meeting\_attendees\_ibfk\_1,

DROP FOREIGN KEY meeting\_attendees\_ibfk\_2;

ALTER TABLE meeting\_attendees

ADD CONSTRAINT fk\_meeting\_attendees\_meeting FOREIGN KEY (meeting\_id)

REFERENCES meetings(meeting\_id) ON DELETE CASCADE;

ALTER TABLE meeting\_attendees

ADD CONSTRAINT fk\_meeting\_attendees\_employee FOREIGN KEY (emp\_id)

REFERENCES employees(emp\_id) ON UPDATE CASCADE;

**Exercise 40: Alter Columns to Add DEFAULT Values and NOT NULL Constraint**

**Exercise:**

1. Create a departments table with the following columns:
   * dept\_id (INT, Primary Key, Auto Increment)
   * dept\_name (VARCHAR(100), Not Null)
   * location (VARCHAR(50))
2. Alter the table to:
   * Add a NOT NULL constraint on the location column.
   * Add a default value of 'Unknown' for the location column.

**Solution:**

-- 1. Create departments table

CREATE TABLE departments (

dept\_id INT AUTO\_INCREMENT PRIMARY KEY,

dept\_name VARCHAR(100) NOT NULL,

location VARCHAR(50)

);

-- 2. Alter the table to add NOT NULL constraint and default value

ALTER TABLE departments

MODIFY location VARCHAR(50) NOT NULL DEFAULT 'Unknown';

**Exercise 41: Add and Drop a Unique Constraint**

**Exercise:**

1. Create a customers table with the following columns:
   * customer\_id (INT, Primary Key, Auto Increment)
   * email (VARCHAR(100), Not Null)
   * phone\_number (VARCHAR(15))
2. Alter the table to:
   * Add a **UNIQUE** constraint on the email column.
   * Later, drop the UNIQUE constraint on email.

**Solution:**

-- 1. Create customers table

CREATE TABLE customers (

customer\_id INT AUTO\_INCREMENT PRIMARY KEY,

email VARCHAR(100) NOT NULL,

phone\_number VARCHAR(15)

);

-- 2. Add UNIQUE constraint on email

ALTER TABLE customers

ADD CONSTRAINT unique\_email UNIQUE (email);

-- 3. Drop the UNIQUE constraint

ALTER TABLE customers

DROP INDEX unique\_email;

**Exercise 42: Rename a Table and a Column**

**Exercise:**

1. Create a suppliers table with the following columns:
   * supplier\_id (INT, Primary Key, Auto Increment)
   * supplier\_name (VARCHAR(100), Not Null)
   * supplier\_location (VARCHAR(100))
2. Rename:
   * The table from suppliers to vendor\_suppliers.
   * The column supplier\_location to location.

**Solution:**

-- 1. Create suppliers table

CREATE TABLE suppliers (

supplier\_id INT AUTO\_INCREMENT PRIMARY KEY,

supplier\_name VARCHAR(100) NOT NULL,

supplier\_location VARCHAR(100)

);

-- 2. Rename the table

RENAME TABLE suppliers TO vendor\_suppliers;

-- 3. Rename the column supplier\_location to location

ALTER TABLE vendor\_suppliers

CHANGE supplier\_location location VARCHAR(100);

**Exercise 43: Add and Drop an Index**

**Exercise:**

1. Create a product\_inventory table with the following columns:
   * product\_id (INT, Primary Key, Auto Increment)
   * product\_name (VARCHAR(100), Not Null)
   * price (DECIMAL(10,2))
2. Add an index on the price column to speed up queries, then later drop the index.

**Solution:**

-- 1. Create product\_inventory table

CREATE TABLE product\_inventory (

product\_id INT AUTO\_INCREMENT PRIMARY KEY,

product\_name VARCHAR(100) NOT NULL,

price DECIMAL(10,2)

);

-- 2. Add index on price

CREATE INDEX idx\_price ON product\_inventory(price);

-- 3. Drop the index

DROP INDEX idx\_price ON product\_inventory;

**Exercise 44: Truncate a Table**

**Exercise:**

1. Create a project\_logs table with the following columns:
   * log\_id (INT, Primary Key, Auto Increment)
   * project\_id (INT)
   * log\_message (TEXT)
2. Insert a few records into the table. Then, use the TRUNCATE command to remove all the records without deleting the table structure.

**Solution:**

-- 1. Create project\_logs table

CREATE TABLE project\_logs (

log\_id INT AUTO\_INCREMENT PRIMARY KEY,

project\_id INT,

log\_message TEXT

);

-- 2. Insert a few records

INSERT INTO project\_logs (project\_id, log\_message) VALUES

(1, 'Project started'),

(2, 'Project completed'),

(1, 'Initial analysis done');

-- 3. Truncate the table

TRUNCATE TABLE project\_logs;

**Exercise 45: Drop a Column**

**Exercise:**

1. Create a vendor\_payments table with the following columns:
   * payment\_id (INT, Primary Key, Auto Increment)
   * vendor\_id (INT, Not Null)
   * payment\_date (DATE, Not Null)
   * payment\_method (VARCHAR(50))
2. Drop the payment\_method column from the table.

**Solution:**

-- 1. Create vendor\_payments table

CREATE TABLE vendor\_payments (

payment\_id INT AUTO\_INCREMENT PRIMARY KEY,

vendor\_id INT NOT NULL,

payment\_date DATE NOT NULL,

payment\_method VARCHAR(50)

);

-- 2. Drop the payment\_method column

ALTER TABLE vendor\_payments

DROP COLUMN payment\_method;

**Exercise 46: Add Multiple Foreign Keys and Drop One**

**Exercise:**

1. Create a client\_contracts table with the following columns:
   * contract\_id (INT, Primary Key, Auto Increment)
   * client\_id (INT, Foreign Key references clients(client\_id))
   * project\_id (INT, Foreign Key references projects(project\_id))
2. Drop the foreign key constraint on the client\_id column while retaining the project\_id foreign key.

**Solution:**

-- 1. Create client\_contracts table

CREATE TABLE client\_contracts (

contract\_id INT AUTO\_INCREMENT PRIMARY KEY,

client\_id INT,

project\_id INT,

FOREIGN KEY (client\_id) REFERENCES clients(client\_id),

FOREIGN KEY (project\_id) REFERENCES projects(project\_id)

);

-- 2. Drop the foreign key on client\_id

ALTER TABLE client\_contracts

DROP FOREIGN KEY client\_contracts\_ibfk\_1;

**Exercise 47: Alter a Column to Change Data Type and Size**

**Exercise:**

1. Create an orders table with the following columns:
   * order\_id (INT, Primary Key, Auto Increment)
   * customer\_name (VARCHAR(50))
   * order\_total (DECIMAL(10,2))
2. Modify the customer\_name column to increase its size to 100 characters.

**Solution:**

-- 1. Create orders table

CREATE TABLE orders (

order\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_name VARCHAR(50),

order\_total DECIMAL(10,2)

);

-- 2. Modify the customer\_name column size

ALTER TABLE orders

MODIFY customer\_name VARCHAR(100);

**Exercise 48: Drop a Table**

**Exercise:**

1. Create a temp\_data table with the following columns:
   * temp\_id (INT, Primary Key, Auto Increment)
   * data\_value (VARCHAR(100))
2. Drop the entire table using the DROP command.

**Solution:**

-- 1. Create temp\_data table

CREATE TABLE temp\_data (

temp\_id INT AUTO\_INCREMENT PRIMARY KEY,

data\_value VARCHAR(100)

);

-- 2. Drop the table

DROP TABLE temp\_data;

**Exercise 49: Add a Primary Key to an Existing Table**

**Exercise:**

1. Create a sales table with the following columns:
   * sale\_id (INT, Not Null)
   * sale\_amount (DECIMAL(10,2))
2. Alter the table to add a **primary key** on the sale\_id column.

**Solution:**

-- 1. Create sales table

CREATE TABLE sales (

sale\_id INT NOT NULL,

sale\_amount DECIMAL(10,2)

);

-- 2. Add a primary key on sale\_id

ALTER TABLE sales

ADD PRIMARY KEY (sale\_id);

**Exercise 50: Add and Drop a Foreign Key Constraint**

**Exercise:**

1. Create an order\_items table with the following columns:
   * item\_id (INT, Primary Key, Auto Increment)
   * order\_id (INT, Foreign Key references orders(order\_id))
2. Drop the foreign key constraint on the order\_id column.

**Solution:**

-- 1. Create order\_items table

CREATE TABLE order\_items (

item\_id INT AUTO\_INCREMENT PRIMARY KEY,

order\_id INT,

FOREIGN KEY (order\_id) REFERENCES orders(order\_id)

);

-- 2. Drop the foreign key on order\_id

ALTER TABLE order\_items

DROP FOREIGN KEY order\_items\_ibfk\_1;

**Exercise 51: Alter Table to Add and Drop Constraints**

**Exercise:**

1. Create a students table with the following columns:
   * student\_id (INT, Primary Key, Auto Increment)
   * first\_name (VARCHAR(50), Not Null)
   * last\_name (VARCHAR(50))
   * email (VARCHAR(100))
2. Perform the following tasks:
   * Add a UNIQUE constraint on the email column.
   * Add a CHECK constraint to ensure the student\_id is greater than 100.
   * Drop the UNIQUE constraint on email.

**Solution:**

-- 1. Create students table

CREATE TABLE students (

student\_id INT AUTO\_INCREMENT PRIMARY KEY,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50),

email VARCHAR(100)

);

-- 2. Add UNIQUE constraint on email

ALTER TABLE students

ADD CONSTRAINT unique\_email UNIQUE (email);

-- 3. Add CHECK constraint to ensure student\_id > 100

ALTER TABLE students

ADD CONSTRAINT chk\_student\_id CHECK (student\_id > 100);

-- 4. Drop UNIQUE constraint on email

ALTER TABLE students

DROP INDEX unique\_email;

**Exercise 52: Alter Multiple Columns and Add Foreign Keys**

**Exercise:**

1. Create a books table with the following columns:
   * book\_id (INT, Primary Key, Auto Increment)
   * title (VARCHAR(100), Not Null)
   * author\_id (INT)
2. Create an authors table:
   * author\_id (INT, Primary Key, Auto Increment)
   * name (VARCHAR(100), Not Null)
3. Perform the following tasks:
   * Alter the books table to add a foreign key reference to authors.
   * Modify the title column to have a new size of 150 characters.
   * Add another column published\_year to books with the data type YEAR.

**Solution:**

-- 1. Create authors table

CREATE TABLE authors (

author\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL

);

-- 2. Create books table

CREATE TABLE books (

book\_id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(100) NOT NULL,

author\_id INT

);

-- 3. Add foreign key for author\_id

ALTER TABLE books

ADD CONSTRAINT fk\_books\_authors FOREIGN KEY (author\_id)

REFERENCES authors(author\_id);

-- 4. Modify title column size to 150 characters

ALTER TABLE books

MODIFY title VARCHAR(150);

-- 5. Add published\_year column

ALTER TABLE books

ADD published\_year YEAR;

**Exercise 53: Rename Multiple Columns and Add Index**

**Exercise:**

1. Create an orders table with the following columns:
   * order\_id (INT, Primary Key, Auto Increment)
   * order\_date (DATE)
   * customer\_name (VARCHAR(100))
2. Perform the following tasks:
   * Rename the customer\_name column to client\_name.
   * Add an INDEX on the order\_date column for faster querying.
   * Rename the table to customer\_orders.

**Solution:**

-- 1. Create orders table

CREATE TABLE orders (

order\_id INT AUTO\_INCREMENT PRIMARY KEY,

order\_date DATE,

customer\_name VARCHAR(100)

);

-- 2. Rename customer\_name to client\_name

ALTER TABLE orders

CHANGE customer\_name client\_name VARCHAR(100);

-- 3. Add INDEX on order\_date

CREATE INDEX idx\_order\_date ON orders(order\_date);

-- 4. Rename table to customer\_orders

RENAME TABLE orders TO customer\_orders;

**Exercise 54: Truncate and Drop Table**

**Exercise:**

1. Create a temp\_sales table with the following columns:
   * temp\_id (INT, Primary Key, Auto Increment)
   * temp\_value (INT)
2. Insert some data into the table. Then:
   * Truncate the temp\_sales table.
   * Drop the temp\_sales table completely.

**Solution:**

-- 1. Create temp\_sales table

CREATE TABLE temp\_sales (

temp\_id INT AUTO\_INCREMENT PRIMARY KEY,

temp\_value INT

);

-- 2. Insert sample data

INSERT INTO temp\_sales (temp\_value) VALUES (100), (200), (300);

-- 3. Truncate the table

TRUNCATE TABLE temp\_sales;

-- 4. Drop the table

DROP TABLE temp\_sales;

**Exercise 55: Add and Drop Composite Primary Key**

**Exercise:**

1. Create a student\_courses table with the following columns:
   * student\_id (INT, Not Null)
   * course\_id (INT, Not Null)
   * grade (CHAR(1))
2. Perform the following tasks:
   * Add a **composite primary key** on the student\_id and course\_id columns.
   * Drop the composite primary key.

**Solution:**

-- 1. Create student\_courses table

CREATE TABLE student\_courses (

student\_id INT NOT NULL,

course\_id INT NOT NULL,

grade CHAR(1)

);

-- 2. Add composite primary key on student\_id and course\_id

ALTER TABLE student\_courses

ADD PRIMARY KEY (student\_id, course\_id);

-- 3. Drop the composite primary key

ALTER TABLE student\_courses

DROP PRIMARY KEY;

**Exercise 56: Add Multiple Foreign Keys and Drop One**

**Exercise:**

1. Create a task\_assignments table with the following columns:
   * assignment\_id (INT, Primary Key, Auto Increment)
   * task\_id (INT, Foreign Key references tasks(task\_id))
   * employee\_id (INT, Foreign Key references employees(emp\_id))
2. Drop the foreign key on task\_id while retaining the foreign key on employee\_id.

**Solution:**

-- 1. Create task\_assignments table

CREATE TABLE task\_assignments (

assignment\_id INT AUTO\_INCREMENT PRIMARY KEY,

task\_id INT,

employee\_id INT,

FOREIGN KEY (task\_id) REFERENCES tasks(task\_id),

FOREIGN KEY (employee\_id) REFERENCES employees(emp\_id)

);

-- 2. Drop the foreign key on task\_id

ALTER TABLE task\_assignments

DROP FOREIGN KEY task\_assignments\_ibfk\_1;

**Exercise 57: Alter Multiple Columns in One Statement**

**Exercise:**

1. Create an invoices table with the following columns:
   * invoice\_id (INT, Primary Key, Auto Increment)
   * amount (DECIMAL(10, 2))
   * due\_date (DATE)
2. Alter the table to:
   * Modify the amount column to DECIMAL(12, 2).
   * Change the due\_date column to allow NULL values.

**Solution:**

-- 1. Create invoices table

CREATE TABLE invoices (

invoice\_id INT AUTO\_INCREMENT PRIMARY KEY,

amount DECIMAL(10,2),

due\_date DATE

);

-- 2. Alter multiple columns

ALTER TABLE invoices

MODIFY amount DECIMAL(12,2),

MODIFY due\_date DATE NULL;

**Exercise 58: Drop and Add a New Primary Key**

**Exercise:**

1. Create a suppliers table with the following columns:
   * supplier\_id (INT)
   * supplier\_name (VARCHAR(100))
2. Perform the following tasks:
   * Add a **primary key** on the supplier\_id column.
   * Drop the primary key and add a new composite primary key on both supplier\_id and supplier\_name.

**Solution:**

-- 1. Create suppliers table

CREATE TABLE suppliers (

supplier\_id INT,

supplier\_name VARCHAR(100)

);

-- 2. Add primary key on supplier\_id

ALTER TABLE suppliers

ADD PRIMARY KEY (supplier\_id);

-- 3. Drop the primary key

ALTER TABLE suppliers

DROP PRIMARY KEY;

-- 4. Add composite primary key on supplier\_id and supplier\_name

ALTER TABLE suppliers

ADD PRIMARY KEY (supplier\_id, supplier\_name);

**Exercise 59: Add and Drop a Unique Constraint**

**Exercise:**

1. Create a projects table with the following columns:
   * project\_id (INT, Primary Key, Auto Increment)
   * project\_code (VARCHAR(50))
   * client\_id (INT)
2. Perform the following tasks:
   * Add a **UNIQUE** constraint on the project\_code column.
   * Drop the **UNIQUE** constraint from the project\_code column.

**Solution:**

-- 1. Create projects table

CREATE TABLE projects (

project\_id INT AUTO\_INCREMENT PRIMARY KEY,

project\_code VARCHAR(50),

client\_id INT

);

-- 2. Add UNIQUE constraint on project\_code

ALTER TABLE projects

ADD CONSTRAINT unique\_project\_code UNIQUE (project\_code);

-- 3. Drop UNIQUE constraint on project\_code

ALTER TABLE projects

DROP INDEX unique\_project\_code;

**Exercise 60: Rename Table and Drop a Column**

**Exercise:**

1. Create a payments table with the following columns:
   * payment\_id (INT, Primary Key, Auto Increment)
   * amount (DECIMAL(10, 2))
   * payment\_date (DATE)
   * payment\_status (VARCHAR(20))
2. Perform the following tasks:
   * Rename the table to customer\_payments.
   * Drop the payment\_status column.

**Solution:**

-- 1. Create payments table

CREATE TABLE payments (

payment\_id INT AUTO\_INCREMENT PRIMARY KEY,

amount DECIMAL(10,2),

payment\_date DATE,

payment\_status VARCHAR(20)

);

-- 2. Rename table to customer\_payments

RENAME TABLE payments TO customer\_payments;

-- 3. Drop payment\_status column

ALTER TABLE customer\_payments

DROP COLUMN payment\_status;

**Exercise 61: Alter Table to Add Multiple Constraints and Modify Data Types**

**Exercise:**

1. Create the following tables:
   * employees:
     + emp\_id (INT, Primary Key, Auto Increment)
     + first\_name (VARCHAR(50), Not Null)
     + last\_name (VARCHAR(50))
     + department\_id (INT)
     + email (VARCHAR(100))
   * departments:
     + department\_id (INT, Primary Key, Auto Increment)
     + department\_name (VARCHAR(100), Not Null)
2. Perform the following tasks:
   * Add a **foreign key** constraint between employees.department\_id and departments.department\_id.
   * Add a **UNIQUE** constraint on the employees.email column.
   * Modify the last\_name column in employees to allow a maximum length of 80 characters.
   * Add a **CHECK** constraint that ensures the emp\_id in employees is greater than 1000.

**Solution:**

-- 1. Create departments table

CREATE TABLE departments (

department\_id INT AUTO\_INCREMENT PRIMARY KEY,

department\_name VARCHAR(100) NOT NULL

);

-- 2. Create employees table

CREATE TABLE employees (

emp\_id INT AUTO\_INCREMENT PRIMARY KEY,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50),

department\_id INT,

email VARCHAR(100)

);

-- 3. Add foreign key constraint between employees.department\_id and departments.department\_id

ALTER TABLE employees

ADD CONSTRAINT fk\_department FOREIGN KEY (department\_id)

REFERENCES departments(department\_id);

-- 4. Add UNIQUE constraint on employees.email

ALTER TABLE employees

ADD CONSTRAINT unique\_email UNIQUE (email);

-- 5. Modify last\_name to allow 80 characters

ALTER TABLE employees

MODIFY last\_name VARCHAR(80);

-- 6. Add CHECK constraint to ensure emp\_id > 1000

ALTER TABLE employees

ADD CONSTRAINT chk\_emp\_id CHECK (emp\_id > 1000);

**Exercise 62: Create Complex Tables and Modify Multiple Columns**

**Exercise:**

1. Create the following tables:
   * projects:
     + project\_id (INT, Primary Key, Auto Increment)
     + project\_name (VARCHAR(100))
     + start\_date (DATE)
     + client\_id (INT)
   * clients:
     + client\_id (INT, Primary Key, Auto Increment)
     + client\_name (VARCHAR(100))
2. Perform the following tasks:
   * Add a **foreign key** constraint between projects.client\_id and clients.client\_id.
   * Modify the project\_name column in projects to be of length 150.
   * Add a new column budget to projects with a data type of DECIMAL(15,2).
   * Add a **CHECK** constraint to ensure the budget is greater than 10,000.
   * Modify the start\_date column to allow NULL values.

**Solution:**

-- 1. Create clients table

CREATE TABLE clients (

client\_id INT AUTO\_INCREMENT PRIMARY KEY,

client\_name VARCHAR(100)

);

-- 2. Create projects table

CREATE TABLE projects (

project\_id INT AUTO\_INCREMENT PRIMARY KEY,

project\_name VARCHAR(100),

start\_date DATE,

client\_id INT

);

-- 3. Add foreign key constraint between projects.client\_id and clients.client\_id

ALTER TABLE projects

ADD CONSTRAINT fk\_projects\_clients FOREIGN KEY (client\_id)

REFERENCES clients(client\_id);

-- 4. Modify project\_name to allow 150 characters

ALTER TABLE projects

MODIFY project\_name VARCHAR(150);

-- 5. Add new column budget with DECIMAL type

ALTER TABLE projects

ADD budget DECIMAL(15,2);

-- 6. Add CHECK constraint to ensure budget > 10000

ALTER TABLE projects

ADD CONSTRAINT chk\_budget CHECK (budget > 10000);

-- 7. Modify start\_date to allow NULL values

ALTER TABLE projects

MODIFY start\_date DATE NULL;

**Exercise 63: Create Multiple Tables and Drop Constraints**

**Exercise:**

1. Create the following tables:
   * orders:
     + order\_id (INT, Primary Key, Auto Increment)
     + order\_date (DATE)
     + customer\_id (INT)
   * customers:
     + customer\_id (INT, Primary Key, Auto Increment)
     + customer\_name (VARCHAR(100))
   * products:
     + product\_id (INT, Primary Key, Auto Increment)
     + product\_name (VARCHAR(100))
     + price (DECIMAL(10, 2))
   * order\_details:
     + order\_id (INT, Foreign Key references orders)
     + product\_id (INT, Foreign Key references products)
     + quantity (INT)
2. Perform the following tasks:
   * Drop the **foreign key** constraint between order\_details.order\_id and orders.order\_id.
   * Drop the **foreign key** constraint between order\_details.product\_id and products.product\_id.
   * Drop the order\_details table.

**Solution:**

-- 1. Create customers table

CREATE TABLE customers (

customer\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_name VARCHAR(100)

);

-- 2. Create orders table

CREATE TABLE orders (

order\_id INT AUTO\_INCREMENT PRIMARY KEY,

order\_date DATE,

customer\_id INT

);

-- 3. Create products table

CREATE TABLE products (

product\_id INT AUTO\_INCREMENT PRIMARY KEY,

product\_name VARCHAR(100),

price DECIMAL(10, 2)

);

-- 4. Create order\_details table

CREATE TABLE order\_details (

order\_id INT,

product\_id INT,

quantity INT,

FOREIGN KEY (order\_id) REFERENCES orders(order\_id),

FOREIGN KEY (product\_id) REFERENCES products(product\_id)

);

-- 5. Drop foreign key constraint between order\_details.order\_id and orders.order\_id

ALTER TABLE order\_details

DROP FOREIGN KEY order\_details\_ibfk\_1;

-- 6. Drop foreign key constraint between order\_details.product\_id and products.product\_id

ALTER TABLE order\_details

DROP FOREIGN KEY order\_details\_ibfk\_2;

-- 7. Drop the order\_details table

DROP TABLE order\_details;

**Exercise 64: Rename Table, Alter Multiple Columns, and Add Constraints**

**Exercise:**

1. Create the following table:
   * payments:
     + payment\_id (INT, Primary Key, Auto Increment)
     + amount (DECIMAL(10, 2))
     + payment\_date (DATE)
     + status (VARCHAR(20))
2. Perform the following tasks:
   * Rename the table to customer\_payments.
   * Add a new column customer\_id (INT).
   * Add a **foreign key** constraint between customer\_payments.customer\_id and customers.customer\_id.
   * Modify the amount column to allow up to 12 digits.
   * Add a **CHECK** constraint to ensure the amount is greater than 0.

**Solution:**

-- 1. Create payments table

CREATE TABLE payments (

payment\_id INT AUTO\_INCREMENT PRIMARY KEY,

amount DECIMAL(10,2),

payment\_date DATE,

status VARCHAR(20)

);

-- 2. Rename table to customer\_payments

RENAME TABLE payments TO customer\_payments;

-- 3. Add customer\_id column

ALTER TABLE customer\_payments

ADD customer\_id INT;

-- 4. Add foreign key constraint between customer\_payments.customer\_id and customers.customer\_id

ALTER TABLE customer\_payments

ADD CONSTRAINT fk\_customer FOREIGN KEY (customer\_id)

REFERENCES customers(customer\_id);

-- 5. Modify amount column to DECIMAL(12,2)

ALTER TABLE customer\_payments

MODIFY amount DECIMAL(12,2);

-- 6. Add CHECK constraint to ensure amount > 0

ALTER TABLE customer\_payments

ADD CONSTRAINT chk\_amount CHECK (amount > 0);

**Exercise 65: Composite Foreign Keys and Renaming Columns**

**Exercise:**

1. Create the following tables:
   * tasks:
     + task\_id (INT, Primary Key, Auto Increment)
     + task\_name (VARCHAR(100))
     + assigned\_to (INT, Foreign Key referencing employees.emp\_id)
   * subtasks:
     + subtask\_id (INT, Primary Key, Auto Increment)
     + task\_id (INT, Foreign Key referencing tasks.task\_id)
     + subtask\_name (VARCHAR(100))
     + subtask\_assigned\_to (INT, Foreign Key referencing employees.emp\_id)
2. Perform the following tasks:
   * Add a **composite foreign key** between subtasks.task\_id and tasks.task\_id and subtasks.subtask\_assigned\_to and employees.emp\_id.
   * Rename the task\_name column in tasks to task\_description.

Solution:

-- 1. Create employees table

CREATE TABLE employees (

emp\_id INT AUTO\_INCREMENT PRIMARY KEY,

emp\_name VARCHAR(100)

);

-- 2. Create tasks table

CREATE TABLE tasks (

task\_id INT AUTO\_INCREMENT PRIMARY KEY,

task\_name VARCHAR(100),

assigned\_to INT,

FOREIGN KEY (assigned\_to) REFERENCES employees(emp\_id)

);

-- 3. Create subtasks table

CREATE TABLE subtasks (

subtask\_id INT AUTO\_INCREMENT PRIMARY KEY,

task\_id INT,

subtask\_name VARCHAR(100),

subtask\_assigned\_to INT,

FOREIGN KEY (task\_id) REFERENCES tasks(task\_id),

FOREIGN KEY (subtask\_assigned\_to) REFERENCES employees(emp\_id)

);

-- 4. Add composite foreign key on subtasks

ALTER TABLE subtasks

ADD CONSTRAINT fk\_subtask\_task FOREIGN KEY (task\_id)

REFERENCES tasks(task\_id),

ADD CONSTRAINT fk\_subtask\_assigned\_to FOREIGN KEY (subtask\_assigned\_to)

REFERENCES employees(emp\_id);

-- 5. Rename task\_name column in tasks to task\_description

ALTER TABLE tasks

CHANGE task\_name task\_description VARCHAR(100);

**Exercise 66: Project Management Schema**

**Exercise:**

1. Create the following tables:
   * **projects**:
     + project\_id (INT, Primary Key, Auto Increment)
     + project\_name (VARCHAR(100))
     + project\_manager (INT, Foreign Key referencing employees.emp\_id)
   * **tasks**:
     + task\_id (INT, Primary Key, Auto Increment)
     + task\_name (VARCHAR(100))
     + project\_id (INT, Foreign Key referencing projects.project\_id)
     + assigned\_to (INT, Foreign Key referencing employees.emp\_id)
2. Perform the following tasks:
   * Add a composite foreign key between tasks.project\_id and projects.project\_id.
   * Rename the project\_name column in projects to name.

**Solution:**

-- 1. Create employees table

CREATE TABLE employees (

emp\_id INT AUTO\_INCREMENT PRIMARY KEY,

emp\_name VARCHAR(100)

);

-- 2. Create projects table

CREATE TABLE projects (

project\_id INT AUTO\_INCREMENT PRIMARY KEY,

project\_name VARCHAR(100),

project\_manager INT,

FOREIGN KEY (project\_manager) REFERENCES employees(emp\_id)

);

-- 3. Create tasks table

CREATE TABLE tasks (

task\_id INT AUTO\_INCREMENT PRIMARY KEY,

task\_name VARCHAR(100),

project\_id INT,

assigned\_to INT,

FOREIGN KEY (project\_id) REFERENCES projects(project\_id),

FOREIGN KEY (assigned\_to) REFERENCES employees(emp\_id)

);

-- 4. Add composite foreign key on tasks

ALTER TABLE tasks

ADD CONSTRAINT fk\_task\_project FOREIGN KEY (project\_id)

REFERENCES projects(project\_id);

-- 5. Rename project\_name column in projects to name

ALTER TABLE projects

CHANGE project\_name name VARCHAR(100);

**Exercise 67: School Database Schema**

**Exercise:**

1. Create the following tables:
   * **students**:
     + student\_id (INT, Primary Key, Auto Increment)
     + student\_name (VARCHAR(100))
     + class\_id (INT, Foreign Key referencing classes.class\_id)
   * **classes**:
     + class\_id (INT, Primary Key, Auto Increment)
     + class\_name (VARCHAR(100))
   * **enrollments**:
     + enrollment\_id (INT, Primary Key, Auto Increment)
     + student\_id (INT, Foreign Key referencing students.student\_id)
     + class\_id (INT, Foreign Key referencing classes.class\_id)
2. Perform the following tasks:
   * Add a composite foreign key between enrollments.student\_id and students.student\_id, and enrollments.class\_id and classes.class\_id.
   * Rename the class\_name column in classes to name.

**Solution:**

-- 1. Create classes table

CREATE TABLE classes (

class\_id INT AUTO\_INCREMENT PRIMARY KEY,

class\_name VARCHAR(100)

);

-- 2. Create students table

CREATE TABLE students (

student\_id INT AUTO\_INCREMENT PRIMARY KEY,

student\_name VARCHAR(100),

class\_id INT,

FOREIGN KEY (class\_id) REFERENCES classes(class\_id)

);

-- 3. Create enrollments table

CREATE TABLE enrollments (

enrollment\_id INT AUTO\_INCREMENT PRIMARY KEY,

student\_id INT,

class\_id INT,

FOREIGN KEY (student\_id) REFERENCES students(student\_id),

FOREIGN KEY (class\_id) REFERENCES classes(class\_id)

);

-- 4. Add composite foreign key on enrollments

ALTER TABLE enrollments

ADD CONSTRAINT fk\_enrollment\_student FOREIGN KEY (student\_id)

REFERENCES students(student\_id),

ADD CONSTRAINT fk\_enrollment\_class FOREIGN KEY (class\_id)

REFERENCES classes(class\_id);

-- 5. Rename class\_name column in classes to name

ALTER TABLE classes

CHANGE class\_name name VARCHAR(100);

**Exercise 68: E-commerce Order System**

**Exercise:**

1. Create the following tables:
   * **customers**:
     + customer\_id (INT, Primary Key, Auto Increment)
     + customer\_name (VARCHAR(100))
   * **orders**:
     + order\_id (INT, Primary Key, Auto Increment)
     + customer\_id (INT, Foreign Key referencing customers.customer\_id)
     + order\_date (DATE)
   * **order\_items**:
     + order\_item\_id (INT, Primary Key, Auto Increment)
     + order\_id (INT, Foreign Key referencing orders.order\_id)
     + product\_name (VARCHAR(100))
2. Perform the following tasks:
   * Add a composite foreign key between order\_items.order\_id and orders.order\_id.
   * Rename the customer\_name column in customers to name.

**Solution:**

-- 1. Create customers table

CREATE TABLE customers (

customer\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_name VARCHAR(100)

);

-- 2. Create orders table

CREATE TABLE orders (

order\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_id INT,

order\_date DATE,

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)

);

-- 3. Create order\_items table

CREATE TABLE order\_items (

order\_item\_id INT AUTO\_INCREMENT PRIMARY KEY,

order\_id INT,

product\_name VARCHAR(100),

FOREIGN KEY (order\_id) REFERENCES orders(order\_id)

);

-- 4. Add composite foreign key on order\_items

ALTER TABLE order\_items

ADD CONSTRAINT fk\_order\_item\_order FOREIGN KEY (order\_id)

REFERENCES orders(order\_id);

-- 5. Rename customer\_name column in customers to name

ALTER TABLE customers

CHANGE customer\_name name VARCHAR(100);

**Exercise 69: Library Management System**

**Exercise:**

1. Create the following tables:
   * **books**:
     + book\_id (INT, Primary Key, Auto Increment)
     + title (VARCHAR(100))
     + author\_id (INT, Foreign Key referencing authors.author\_id)
   * **authors**:
     + author\_id (INT, Primary Key, Auto Increment)
     + author\_name (VARCHAR(100))
   * **loans**:
     + loan\_id (INT, Primary Key, Auto Increment)
     + book\_id (INT, Foreign Key referencing books.book\_id)
     + borrower\_id (INT, Foreign Key referencing borrowers.borrower\_id)
   * **borrowers**:
     + borrower\_id (INT, Primary Key, Auto Increment)
     + borrower\_name (VARCHAR(100))
2. Perform the following tasks:
   * Add a composite foreign key between loans.book\_id and books.book\_id, and loans.borrower\_id and borrowers.borrower\_id.
   * Rename the title column in books to book\_title.

**Solution:**

-- 1. Create authors table

CREATE TABLE authors (

author\_id INT AUTO\_INCREMENT PRIMARY KEY,

author\_name VARCHAR(100)

);

-- 2. Create books table

CREATE TABLE books (

book\_id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(100),

author\_id INT,

FOREIGN KEY (author\_id) REFERENCES authors(author\_id)

);

-- 3. Create borrowers table

CREATE TABLE borrowers (

borrower\_id INT AUTO\_INCREMENT PRIMARY KEY,

borrower\_name VARCHAR(100)

);

-- 4. Create loans table

CREATE TABLE loans (

loan\_id INT AUTO\_INCREMENT PRIMARY KEY,

book\_id INT,

borrower\_id INT,

FOREIGN KEY (book\_id) REFERENCES books(book\_id),

FOREIGN KEY (borrower\_id) REFERENCES borrowers(borrower\_id)

);

-- 5. Add composite foreign key on loans

ALTER TABLE loans

ADD CONSTRAINT fk\_loan\_book FOREIGN KEY (book\_id)

REFERENCES books(book\_id),

ADD CONSTRAINT fk\_loan\_borrower FOREIGN KEY (borrower\_id)

REFERENCES borrowers(borrower\_id);

-- 6. Rename title column in books to book\_title

ALTER TABLE books

CHANGE title book\_title VARCHAR(100);

**Exercise 70: Inventory Management System**

**Exercise:**

1. Create the following tables:
   * **categories**:
     + category\_id (INT, Primary Key, Auto Increment)
     + category\_name (VARCHAR(100), UNIQUE)
   * **products**:
     + product\_id (INT, Primary Key, Auto Increment)
     + product\_name (VARCHAR(100))
     + category\_id (INT, Foreign Key referencing categories.category\_id)
     + stock\_quantity (INT CHECK (stock\_quantity >= 0))
     + price (DECIMAL(10, 2) CHECK (price >= 0))
2. Perform the following tasks:
   * Add a UNIQUE constraint on product\_name in products.
   * Rename category\_name in categories to name.
   * Add a new column supplier\_id to products (INT).
   * Alter the products table to set supplier\_id as NOT NULL.

**Solution:**

-- 1. Create categories table

CREATE TABLE categories (

category\_id INT AUTO\_INCREMENT PRIMARY KEY,

category\_name VARCHAR(100) UNIQUE

);

-- 2. Create products table

CREATE TABLE products (

product\_id INT AUTO\_INCREMENT PRIMARY KEY,

product\_name VARCHAR(100),

category\_id INT,

stock\_quantity INT CHECK (stock\_quantity >= 0),

price DECIMAL(10, 2) CHECK (price >= 0),

FOREIGN KEY (category\_id) REFERENCES categories(category\_id)

);

-- 3. Add UNIQUE constraint on product\_name

ALTER TABLE products

ADD CONSTRAINT uq\_product\_name UNIQUE (product\_name);

-- 4. Rename category\_name column in categories to name

ALTER TABLE categories

CHANGE category\_name name VARCHAR(100);

-- 5. Add new column supplier\_id to products

ALTER TABLE products

ADD supplier\_id INT;

-- 6. Alter supplier\_id to be NOT NULL

ALTER TABLE products

MODIFY supplier\_id INT NOT NULL;

**Exercise 71: Hotel Reservation System**

**Exercise:**

1. Create the following tables:
   * **hotels**:
     + hotel\_id (INT, Primary Key, Auto Increment)
     + hotel\_name (VARCHAR(100), UNIQUE)
     + location (VARCHAR(100))
   * **rooms**:
     + room\_id (INT, Primary Key, Auto Increment)
     + hotel\_id (INT, Foreign Key referencing hotels.hotel\_id)
     + room\_number (VARCHAR(10))
     + room\_type (VARCHAR(50))
     + price (DECIMAL(10, 2))
2. Perform the following tasks:
   * Add a composite unique constraint on hotel\_id and room\_number in rooms.
   * Rename room\_type in rooms to type.
   * Add a CHECK constraint on price to ensure it is positive.

**Solution:**

-- 1. Create hotels table

CREATE TABLE hotels (

hotel\_id INT AUTO\_INCREMENT PRIMARY KEY,

hotel\_name VARCHAR(100) UNIQUE,

location VARCHAR(100)

);

-- 2. Create rooms table

CREATE TABLE rooms (

room\_id INT AUTO\_INCREMENT PRIMARY KEY,

hotel\_id INT,

room\_number VARCHAR(10),

room\_type VARCHAR(50),

price DECIMAL(10, 2),

FOREIGN KEY (hotel\_id) REFERENCES hotels(hotel\_id)

);

-- 3. Add composite unique constraint on hotel\_id and room\_number

ALTER TABLE rooms

ADD CONSTRAINT uq\_hotel\_room UNIQUE (hotel\_id, room\_number);

-- 4. Rename room\_type column in rooms to type

ALTER TABLE rooms

CHANGE room\_type type VARCHAR(50);

-- 5. Add CHECK constraint on price

ALTER TABLE rooms

ADD CONSTRAINT chk\_price CHECK (price > 0);

**Exercise 72: Customer Feedback System**

**Exercise:**

1. Create the following tables:
   * **feedback\_types**:
     + feedback\_type\_id (INT, Primary Key, Auto Increment)
     + type\_name (VARCHAR(100), UNIQUE)
   * **feedbacks**:
     + feedback\_id (INT, Primary Key, Auto Increment)
     + customer\_name (VARCHAR(100))
     + feedback\_type\_id (INT, Foreign Key referencing feedback\_types.feedback\_type\_id)
     + comments (TEXT)
     + feedback\_date (DATE)
2. Perform the following tasks:
   * Add a foreign key constraint on feedback\_type\_id.
   * Rename customer\_name in feedbacks to name.
   * Create an index on feedback\_date in feedbacks.

**Solution:**

-- 1. Create feedback\_types table

CREATE TABLE feedback\_types (

feedback\_type\_id INT AUTO\_INCREMENT PRIMARY KEY,

type\_name VARCHAR(100) UNIQUE

);

-- 2. Create feedbacks table

CREATE TABLE feedbacks (

feedback\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_name VARCHAR(100),

feedback\_type\_id INT,

comments TEXT,

feedback\_date DATE,

FOREIGN KEY (feedback\_type\_id) REFERENCES feedback\_types(feedback\_type\_id)

);

-- 3. Rename customer\_name column in feedbacks to name

ALTER TABLE feedbacks

CHANGE customer\_name name VARCHAR(100);

-- 4. Create index on feedback\_date

CREATE INDEX idx\_feedback\_date ON feedbacks(feedback\_date);

**Exercise 73: Employee Performance Tracking**

**Exercise:**

1. Create the following tables:
   * **departments**:
     + department\_id (INT, Primary Key, Auto Increment)
     + department\_name (VARCHAR(100), UNIQUE)
   * **employees**:
     + employee\_id (INT, Primary Key, Auto Increment)
     + name (VARCHAR(100))
     + department\_id (INT, Foreign Key referencing departments.department\_id)
     + hire\_date (DATE)
     + salary (DECIMAL(10, 2))
   * **performance\_reviews**:
     + review\_id (INT, Primary Key, Auto Increment)
     + employee\_id (INT, Foreign Key referencing employees.employee\_id)
     + review\_date (DATE)
     + score (INT CHECK (score BETWEEN 1 AND 5))
2. Perform the following tasks:
   * Add a composite foreign key between performance\_reviews.employee\_id and employees.employee\_id.
   * Rename score in performance\_reviews to rating.
   * Add a CHECK constraint on salary to ensure it is greater than or equal to the minimum wage (assume 1000).

**Solution:**

-- 1. Create departments table

CREATE TABLE departments (

department\_id INT AUTO\_INCREMENT PRIMARY KEY,

department\_name VARCHAR(100) UNIQUE

);

-- 2. Create employees table

CREATE TABLE employees (

employee\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100),

department\_id INT,

hire\_date DATE,

salary DECIMAL(10, 2),

FOREIGN KEY (department\_id) REFERENCES departments(department\_id)

);

-- 3. Create performance\_reviews table

CREATE TABLE performance\_reviews (

review\_id INT AUTO\_INCREMENT PRIMARY KEY,

employee\_id INT,

review\_date DATE,

score INT CHECK (score BETWEEN 1 AND 5),

FOREIGN KEY (employee\_id) REFERENCES employees(employee\_id)

);

-- 4. Add composite foreign key on performance\_reviews

ALTER TABLE performance\_reviews

ADD CONSTRAINT fk\_review\_employee FOREIGN KEY (employee\_id)

REFERENCES employees(employee\_id);

-- 5. Rename score column in performance\_reviews to rating

ALTER TABLE performance\_reviews

CHANGE score rating INT CHECK (rating BETWEEN 1 AND 5);

-- 6. Add CHECK constraint on salary

ALTER TABLE employees

ADD CONSTRAINT chk\_salary CHECK (salary >= 1000);

**Exercise 74: Conference Management System**

**Exercise:**

1. Create the following tables:
   * **conferences**:
     + conference\_id (INT, Primary Key, Auto Increment)
     + conference\_name (VARCHAR(100), UNIQUE)
     + date (DATE)
   * **attendees**:
     + attendee\_id (INT, Primary Key, Auto Increment)
     + attendee\_name (VARCHAR(100))
   * **registrations**:
     + registration\_id (INT, Primary Key, Auto Increment)
     + conference\_id (INT, Foreign Key referencing conferences.conference\_id)
     + attendee\_id (INT, Foreign Key referencing attendees.attendee\_id)
2. Perform the following tasks:
   * Add a composite foreign key between registrations.conference\_id and conferences.conference\_id, and registrations.attendee\_id and attendees.attendee\_id.
   * Rename conference\_name in conferences to name.
   * Create an index on date in conferences.

**Solution:**

-- 1. Create conferences table

CREATE TABLE conferences (

conference\_id INT AUTO\_INCREMENT PRIMARY KEY,

conference\_name VARCHAR(100) UNIQUE,

date DATE

);

-- 2. Create attendees table

CREATE TABLE attendees (

attendee\_id INT AUTO\_INCREMENT PRIMARY KEY,

attendee\_name VARCHAR(100)

);

-- 3. Create registrations table

CREATE TABLE registrations (

registration\_id INT AUTO\_INCREMENT PRIMARY KEY,

conference\_id INT,

attendee\_id INT,

FOREIGN KEY (conference\_id) REFERENCES conferences(conference\_id),

FOREIGN KEY (attendee\_id) REFERENCES attendees(attendee\_id)

);

-- 4. Rename conference\_name column in conferences to name

ALTER TABLE conferences

CHANGE conference\_name name VARCHAR(100);

-- 5. Create index on date

CREATE INDEX idx\_conference\_date ON conferences(date);

**Exercise 75: Online Learning Platform**

**Exercise:**

1. Create the following tables:
   * **courses**:
     + course\_id (INT, Primary Key, Auto Increment)
     + course\_title (VARCHAR(100), UNIQUE)
     + duration (INT CHECK (duration > 0)) -- duration in hours
   * **students**:
     + student\_id (INT, Primary Key, Auto Increment)
     + student\_name (VARCHAR(100))
   * **enrollments**:
     + enrollment\_id (INT, Primary Key, Auto Increment)
     + course\_id (INT, Foreign Key referencing courses.course\_id)
     + student\_id (INT, Foreign Key referencing students.student\_id)
     + enrollment\_date (DATE)
2. Perform the following tasks:
   * Add a composite foreign key between enrollments.course\_id and courses.course\_id, and enrollments.student\_id and students.student\_id.
   * Rename course\_title in courses to title.
   * Create a CHECK constraint on duration to ensure it is at least 1 hour.

**Solution:**

-- 1. Create courses table

CREATE TABLE courses (

course\_id INT AUTO\_INCREMENT PRIMARY KEY,

course\_title VARCHAR(100) UNIQUE,

duration INT CHECK (duration > 0)

);

-- 2. Create students table

CREATE TABLE students (

student\_id INT AUTO\_INCREMENT PRIMARY KEY,

student\_name VARCHAR(100)

);

-- 3. Create enrollments table

CREATE TABLE enrollments (

enrollment\_id INT AUTO\_INCREMENT PRIMARY KEY,

course\_id INT,

student\_id INT,

enrollment\_date DATE,

FOREIGN KEY (course\_id) REFERENCES courses(course\_id),

FOREIGN KEY (student\_id) REFERENCES students(student\_id)

);

-- 4. Rename course\_title column in courses to title

ALTER TABLE courses

CHANGE course\_title title VARCHAR(100);

-- 5. Add CHECK constraint on duration

ALTER TABLE courses

ADD CONSTRAINT chk\_duration CHECK (duration >= 1);

**Exercise 76: Fitness Center Management**

**Exercise:**

1. Create the following tables:
   * **members**:
     + member\_id (INT, Primary Key, Auto Increment)
     + member\_name (VARCHAR(100))
     + membership\_start\_date (DATE)
     + membership\_type (VARCHAR(50))
   * **trainers**:
     + trainer\_id (INT, Primary Key, Auto Increment)
     + trainer\_name (VARCHAR(100))
     + specialization (VARCHAR(100))
   * **sessions**:
     + session\_id (INT, Primary Key, Auto Increment)
     + session\_date (DATE)
     + trainer\_id (INT, Foreign Key referencing trainers.trainer\_id)
     + member\_id (INT, Foreign Key referencing members.member\_id)
     + session\_duration (INT CHECK (session\_duration > 0)) -- duration in minutes
2. Perform the following tasks:
   * Add a UNIQUE constraint on membership\_type in members.
   * Rename specialization in trainers to expertise.
   * Create an index on session\_date in sessions.

**Solution:**

-- 1. Create members table

CREATE TABLE members (

member\_id INT AUTO\_INCREMENT PRIMARY KEY,

member\_name VARCHAR(100),

membership\_start\_date DATE,

membership\_type VARCHAR(50)

);

-- 2. Create trainers table

CREATE TABLE trainers (

trainer\_id INT AUTO\_INCREMENT PRIMARY KEY,

trainer\_name VARCHAR(100),

specialization VARCHAR(100)

);

-- 3. Create sessions table

CREATE TABLE sessions (

session\_id INT AUTO\_INCREMENT PRIMARY KEY,

session\_date DATE,

trainer\_id INT,

member\_id INT,

session\_duration INT CHECK (session\_duration > 0),

FOREIGN KEY (trainer\_id) REFERENCES trainers(trainer\_id),

FOREIGN KEY (member\_id) REFERENCES members(member\_id)

);

-- 4. Add UNIQUE constraint on membership\_type

ALTER TABLE members

ADD CONSTRAINT uq\_membership\_type UNIQUE (membership\_type);

-- 5. Rename specialization column in trainers to expertise

ALTER TABLE trainers

CHANGE specialization expertise VARCHAR(100);

-- 6. Create index on session\_date

CREATE INDEX idx\_session\_date ON sessions(session\_date);

**Exercise 77: E-commerce Order Management**

**Exercise:**

1. Create the following tables:
   * **customers**:
     + customer\_id (INT, Primary Key, Auto Increment)
     + customer\_name (VARCHAR(100))
     + email (VARCHAR(100) UNIQUE)
   * **orders**:
     + order\_id (INT, Primary Key, Auto Increment)
     + customer\_id (INT, Foreign Key referencing customers.customer\_id)
     + order\_date (DATE)
     + total\_amount (DECIMAL(10, 2) CHECK (total\_amount > 0))
   * **order\_items**:
     + order\_item\_id (INT, Primary Key, Auto Increment)
     + order\_id (INT, Foreign Key referencing orders.order\_id)
     + product\_name (VARCHAR(100))
     + quantity (INT CHECK (quantity > 0))
     + price (DECIMAL(10, 2) CHECK (price >= 0))
2. Perform the following tasks:
   * Add a composite foreign key between order\_items.order\_id and orders.order\_id.
   * Rename total\_amount in orders to amount\_due.
   * Create an index on customer\_id in orders.

**Solution:**

-- 1. Create customers table

CREATE TABLE customers (

customer\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_name VARCHAR(100),

email VARCHAR(100) UNIQUE

);

-- 2. Create orders table

CREATE TABLE orders (

order\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_id INT,

order\_date DATE,

total\_amount DECIMAL(10, 2) CHECK (total\_amount > 0),

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)

);

-- 3. Create order\_items table

CREATE TABLE order\_items (

order\_item\_id INT AUTO\_INCREMENT PRIMARY KEY,

order\_id INT,

product\_name VARCHAR(100),

quantity INT CHECK (quantity > 0),

price DECIMAL(10, 2) CHECK (price >= 0),

FOREIGN KEY (order\_id) REFERENCES orders(order\_id)

);

-- 4. Add composite foreign key on order\_items

ALTER TABLE order\_items

ADD CONSTRAINT fk\_order\_items\_order FOREIGN KEY (order\_id)

REFERENCES orders(order\_id);

-- 5. Rename total\_amount column in orders to amount\_due

ALTER TABLE orders

CHANGE total\_amount amount\_due DECIMAL(10, 2) CHECK (amount\_due > 0);

-- 6. Create index on customer\_id

CREATE INDEX idx\_customer\_id ON orders(customer\_id);

**Exercise 78: Library Management System**

**Exercise:**

1. Create the following tables:
   * **books**:
     + book\_id (INT, Primary Key, Auto Increment)
     + title (VARCHAR(100))
     + author (VARCHAR(100))
     + isbn (VARCHAR(20) UNIQUE)
   * **members**:
     + member\_id (INT, Primary Key, Auto Increment)
     + member\_name (VARCHAR(100))
     + membership\_date (DATE)
   * **loans**:
     + loan\_id (INT, Primary Key, Auto Increment)
     + book\_id (INT, Foreign Key referencing books.book\_id)
     + member\_id (INT, Foreign Key referencing members.member\_id)
     + loan\_date (DATE)
     + return\_date (DATE)
2. Perform the following tasks:
   * Add a CHECK constraint on return\_date to ensure it is greater than or equal to loan\_date.
   * Rename title in books to book\_title.
   * Create an index on membership\_date in members.

**Solution:**

-- 1. Create books table

CREATE TABLE books (

book\_id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(100),

author VARCHAR(100),

isbn VARCHAR(20) UNIQUE

);

-- 2. Create members table

CREATE TABLE members (

member\_id INT AUTO\_INCREMENT PRIMARY KEY,

member\_name VARCHAR(100),

membership\_date DATE

);

-- 3. Create loans table

CREATE TABLE loans (

loan\_id INT AUTO\_INCREMENT PRIMARY KEY,

book\_id INT,

member\_id INT,

loan\_date DATE,

return\_date DATE,

FOREIGN KEY (book\_id) REFERENCES books(book\_id),

FOREIGN KEY (member\_id) REFERENCES members(member\_id),

CHECK (return\_date >= loan\_date)

);

-- 4. Rename title column in books to book\_title

ALTER TABLE books

CHANGE title book\_title VARCHAR(100);

-- 5. Create index on membership\_date

CREATE INDEX idx\_membership\_date ON members(membership\_date);

**Exercise 79: Event Management System**

**Exercise:**

1. Create the following tables:
   * **events**:
     + event\_id (INT, Primary Key, Auto Increment)
     + event\_name (VARCHAR(100) UNIQUE)
     + event\_date (DATE)
   * **participants**:
     + participant\_id (INT, Primary Key, Auto Increment)
     + participant\_name (VARCHAR(100))
   * **registrations**:
     + registration\_id (INT, Primary Key, Auto Increment)
     + event\_id (INT, Foreign Key referencing events.event\_id)
     + participant\_id (INT, Foreign Key referencing participants.participant\_id)
     + registration\_date (DATE)
2. Perform the following tasks:
   * Add a composite foreign key between registrations.event\_id and events.event\_id, and registrations.participant\_id and participants.participant\_id.
   * Rename event\_name in events to name.
   * Create a CHECK constraint on event\_date to ensure it is not in the past.

**Solution:**

-- 1. Create events table

CREATE TABLE events (

event\_id INT AUTO\_INCREMENT PRIMARY KEY,

event\_name VARCHAR(100) UNIQUE,

event\_date DATE CHECK (event\_date >= CURDATE())

);

-- 2. Create participants table

CREATE TABLE participants (

participant\_id INT AUTO\_INCREMENT PRIMARY KEY,

participant\_name VARCHAR(100)

);

-- 3. Create registrations table

CREATE TABLE registrations (

registration\_id INT AUTO\_INCREMENT PRIMARY KEY,

event\_id INT,

participant\_id INT,

registration\_date DATE,

FOREIGN KEY (event\_id) REFERENCES events(event\_id),

FOREIGN KEY (participant\_id) REFERENCES participants(participant\_id)

);

-- 4. Rename event\_name column in events to name

ALTER TABLE events

CHANGE event\_name name VARCHAR(100);

-- 5. Create CHECK constraint on event\_date

ALTER TABLE events

ADD CONSTRAINT chk\_event\_date CHECK (event\_date >= CURDATE());

**Exercise 80: Transportation Management System**

**Exercise:**

1. Create the following tables:
   * **vehicles**:
     + vehicle\_id (INT, Primary Key, Auto Increment)
     + vehicle\_type (VARCHAR(50))
     + license\_plate (VARCHAR(20) UNIQUE)
   * **drivers**:
     + driver\_id (INT, Primary Key, Auto Increment)
     + driver\_name (VARCHAR(100))
     + license\_number (VARCHAR(50) UNIQUE)
   * **trips**:
     + trip\_id (INT, Primary Key, Auto Increment)
     + vehicle\_id (INT, Foreign Key referencing vehicles.vehicle\_id)
     + driver\_id (INT, Foreign Key referencing drivers.driver\_id)
     + trip\_date (DATE)
     + trip\_duration (INT CHECK (trip\_duration > 0)) -- duration in minutes
2. Perform the following tasks:
   * Add a composite foreign key between trips.vehicle\_id and vehicles.vehicle\_id, and trips.driver\_id and drivers.driver\_id.
   * Rename vehicle\_type in vehicles to type.
   * Create an index on trip\_date in trips.

**Solution:**

-- 1. Create vehicles table

CREATE TABLE vehicles (

vehicle\_id INT AUTO\_INCREMENT PRIMARY KEY,

vehicle\_type VARCHAR(50) UNIQUE,

license\_plate VARCHAR(20) UNIQUE

);

-- 2. Create drivers table

CREATE TABLE drivers (

driver\_id INT AUTO\_INCREMENT PRIMARY KEY,

driver\_name VARCHAR(100),

license\_number VARCHAR(50) UNIQUE

);

-- 3. Create trips table

CREATE TABLE trips (

trip\_id INT AUTO\_INCREMENT PRIMARY KEY,

vehicle\_id INT,

driver\_id INT,

trip\_date DATE,

trip\_duration INT CHECK (trip\_duration > 0),

FOREIGN KEY (vehicle\_id) REFERENCES vehicles(vehicle\_id),

FOREIGN KEY (driver\_id) REFERENCES drivers(driver\_id)

);

-- 4. Add composite foreign key on trips

ALTER TABLE trips

ADD CONSTRAINT fk\_trips\_vehicle\_driver FOREIGN KEY (vehicle\_id, driver\_id)

REFERENCES vehicles(vehicle\_id), drivers(driver\_id);

-- 5. Rename vehicle\_type column in vehicles to type

ALTER TABLE vehicles

CHANGE vehicle\_type type VARCHAR(50);

-- 6. Create index on trip\_date

CREATE INDEX idx\_trip\_date ON trips(trip\_date);